WHAT IS CLAIMED IS:

1. A computer system for testing software, comprising:

a scanning engine for creating a dictionary of types found within executable binaries of a build; and

an execution monitor for invoking a method of a type from the dictionary and storing a result returned by invocation of the method in the dictionary.

2. The system of claim 1 further comprising a results manager for comparing results returned by two different invocations of a method.

3. The system of claim 1 further comprising a configurator for adding an article of configuration to a configuration for execution.

4. The system of claim 1 further comprising a configuration loader for loading a configuration for execution.

5. The system of claim 1 further comprising a graphical user interface for viewing a representation of the dictionary.

6. The system of claim 1 further comprising a delta generator for computing a difference between executable binaries of two software builds.

7. The system of claim 1 further comprising a database manager for managing configurations and execution runs.

8. The system of claim 1 further comprising an editor for editing information of a type in the dictionary.

9. The system of claim 1 wherein the scanning engine comprises a dependency manager for tracking type dependencies.

10. The system of claim 5 wherein the graphical user interface comprises a query builder for creating a query about a type in the dictionary.

11. A method for testing software in a computer system, comprising the steps of:

scanning a binary executable from a build;

creating a dictionary of types found in the build from scanning the binary executable;

invoking a method of a type in the dictionary;

storing the results of the invoked method in the dictionary; and

comparing the results of the invocation of the method with the results of an invocation of the method from another build.

12. The method of claim 11 further comprising the step of adding an article of configuration to the dictionary.

13. The method of claim 11 further comprising the step of loading configuration information for a type in the dictionary.

14. The method of claim 11 further comprising the step of tracking the dependencies of a type found in the build.

15. The method of claim 11 further comprising the step of storing a member of a type in the dictionary.

16. The method of claim 11 further comprising the step of editing a type in the dictionary.

17. The method of claim 11 further comprising the step of creating an instance of a type from the dictionary for invocation.

18. The method of claim 11 further comprising the step of generating a listing of code that produced different results from the results of the other build.

19. The method of claim 11 further comprising the step of reporting the number of blocks of code that executed during invocation of the method.

20. The method of claim 11 further comprising the step of reporting the number of branches from blocks of code that executed during invocation of the method.

21. The method of claim 11 further comprising the step of reporting the number of functions in the code that executed during invocation of the method.

22. The method of claim 11 further comprising the step of reporting the number of source files for the code that executed during invocation of the method.

23. The method of claim 11 further comprising the step of reporting the number of classes in the code that executed during invocation of the method.

24. The method of claim 15 wherein the step of storing members of a type in the dictionary comprises storing a method of the type.

25. The method of claim 15 wherein the step of storing members of a type in the dictionary comprises storing a parameter of the type.

26. The method of claim 15 wherein the step of storing members of a type in the dictionary comprises storing a validator of the type.

27. The method of claim 15 wherein the step of storing members of a type in the dictionary comprises storing a constructor of the type.

28. The method of claim 11 further comprising the step of generating a report listing any difference between types of the dictionary and types of a dictionary from another build.

29. The method of claim 28 wherein the step of generating a report listing any difference comprises listing any type that is new compared to the dictionary of the other build.

30. The method of claim 28 wherein the step of generating a report listing any difference comprises listing any type that is deleted compared to the dictionary of the other build.

31. A computer-readable medium having stored thereon a data structure, comprising:

a set of elements of a software build of a product;

a subset of the set of elements having information fields for each assembly of the build;

a subset of the set of elements having information fields for each type of an assembly; and

a subset of the set of elements having information for each method of a type; and

a subset of the set of elements having information for an execution result of a method.

32. The data structure of claim 31 further comprising a subset of the set of elements having information for a validator of a type.

33. A computer-readable medium having computer-executable instructions for performing the steps of:

scanning a binary executable from a build;

creating a dictionary of types found in the build from scanning the binary executable;

invoking a method of a type in the dictionary;

comparing a result of the invocation of the method with a result of an invocation of the method from another build; and

executing selective tests for any type that has a result different from the result of an invocation of the method from the other build.

34. A computer-readable medium having computer-executable components, comprising:

a scanning engine for creating a dictionary of types found within executable binaries of a build;

an execution monitor for invoking a method of a type from the dictionary and storing a result returned by invocation of the method in the dictionary; and

a results manager for comparing results returned by two different invocations of a method.

35. A computer system for testing software, comprising:

means for scanning a binary executable from a build;

means for creating a dictionary of types found in the build from scanning the binary executable;

means for invoking a method of a type in the dictionary; and

means for comparing a result of the invocation of the method with a result of an invocation of the method from another build.